

## Procedural Volumetric Cloud Modeling, Animation, and Real-time Techniques, part 2

**David S. Ebert**

*School of Electrical and Computer Engineering  
Purdue University  
[ebertd@purdue.edu](mailto:ebertd@purdue.edu)*

### Stratus And Cirrus Cloud Effects

#### **Stratus Clouds**

- Use a few implicits to specify extent of layer
- Use procedural techniques for details
- Denser and less wispy

#### **Cirrus Clouds**

- Use implicits for each cloud or for global shape
- Thinner, less dense, wispier



### Another Example (Henrik Wann Jensen)

#### *Procedural Cloud Model Based on the Techniques Presented*

- Generates a large number of points describing cloud density

#### *Realistic Cloud and Environmental Illumination Using Photon Maps*

#### *Animation: Little Fluffy Clouds*

- Cloud density is increased procedurally
- Sun rises, cloud layer forms, sun sets

### Examples Using Commercial Systems: A/W Maya

#### *Rendering:*

- Volumetric cloud plug-in

#### *Animation*

- Cloud formation dynamics in MEL

### Volumetric Cloud Plug-in

(Marlin Rowley, Vlad Korolev, David Ebert)

#### *Prototype Volume Rendering Plug-in*

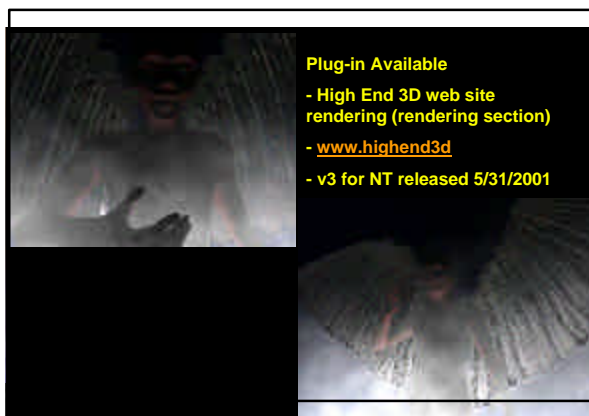
#### *Attached to Volume Light Shape*

#### *Cloud Shape: 3 Spherical Primitives*

#### *4 Cloud Types:*

- Misty
- Cumulus
- Cirrus
- Implicit

### Volumetric Cloud Plug-in: Examples



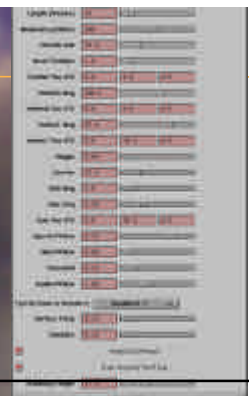
Plug-in Available  
- High End 3D web site  
rendering (rendering section)  
- [www.highend3d](http://www.highend3d)  
- v3 for NT released 5/31/2001

### Cloud Dynamics in MEL (Ruchigartha)

#### *Specialized Particle System*

#### *Dynamics Simulates*

- Buoyant bubbles
- Temperature gradients - controls velocity
- Vortices
- Gravity
- Wind fields



## Cloud Dynamics in MEL: Simulation

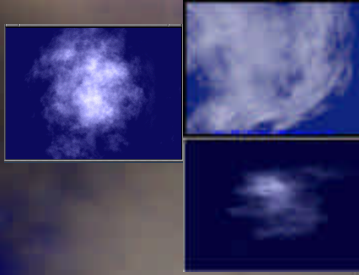
### Particle Emitter

- Numerous settable attributes

### Evaluate Forces on Particles

### Create Children - Split Particles

### Particle Death - Stabilize



## Real-time Dense Gases: Issues

### Volume Rendering vs. Approximations

### Static vs. Dynamic Models

### Semi-transparent Volume Accumulation

### Illumination

### Shadowing

## Issues for Real-time Gases: Volume Approximations

### Particle Systems – Only Practical for Thin Gases

- No inter-particle illumination, shadowing
- Often simple transparency model (or none) – depth sorted?
- Probabilistic shading and shadowing can be used

### Imposters / Billboards – Good for Distant Clouds

- For close-ups and fly-throughs must integrate cloud slabs onto imposter
  - Very time consuming – slows performance
  - Use pre-computed tables to improve performance

## Issues for Real-Time Gases: Volume Approximations (cont.)

### Textured Ellipsoids – Good for Distant Clouds

- Problem 1: need to handle view dependent illumination and shadowing
- Problem 2: fly-throughs
  - must integrate cloud onto plane that slices through ellipsoid
    - Need to update each frame
  - Very time consuming – slows performance
  - Use pre-computed tables to improve performance

## Issues for Real-time Gases: Volume Rendering (Overview)

### Hardware Approaches to Real-Time Volume Rendering

- Mitsubishi VolumePro board (>\$5000)
- 3D texture mapping hardware
  - Nvidia GeForce3, ATI Radeon (< \$400)
  - SGI Octane, Onyx, ... (>\$10,000)
- Limited resolution based on board memory
  - 256<sup>3</sup> (64Mb)?



### Interactive Software Solutions

- Splatting – Comes closest but is still seconds / frame

## Issues for Volumetric Gases: Static Modeling

### 3D Textures for Gas Density

- Limited by resolution of 3D texture: 256<sup>3</sup> (64Mb)
  - Not a very detailed cloud, want 1000<sup>3</sup> at least
  - What about shadow volume, illumination volumes, etc. => even more memory
- Precision of densities / opacities: Is 8 bits enough?

### Global Density Model + Volume Detail Texture (Noise Texture)

- Need dependent texture reads



## Issues for Real-time Volumetric Gases: Dynamic Models

### *Dynamically Change 3D Texture Densities*

- Need ability to update portions of 3D textures at 30 fps

### *Change 3D Texture Indices Algorithmically*

- How quick can you change the texture coordinates on the slices?

### *Use a Changing Smaller Texture to Dynamically Offset the 3D Texture Lookup*

### *Could Generate Geometry on the Fly (Micropolygons)*

- Need capability to generate new triangles at the vertex or fragment processing level
  - E.g. from a vertex program on a Nvidia chip

Can use dummy geometry – but no textures in v.p.

## Issues for Real-time Volumetric Gases: Opacity Accumulation

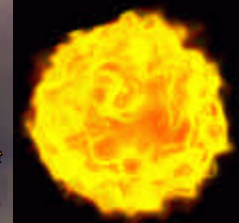
### *Need Exponential Accumulation of Gas Densities:*

$$e^{-t \int_{r1}^{r2} \text{density}(p) dp}$$

- Most systems use simple linear blend

### *Can Pre-integrate Accumulated Opacity Within a Slab and Store That in the Texture (e.g., Engel 2001)*

- Opacities at front and back plus step size become texture coordinates
- Requires dependent texture read



Courtesy of Klaus Engel, Pre-Integrated Volume Renderer V1.7, 15 fps, 2001

## Issues for Real-time Volumetric Gases: Illumination

### *How to Simulate Bi-directional Reflection Function for Low-albedo Illumination*

- 2D texture maps indexed by eye angle and light angle?
  - Needs dependent texture read

### *How to Simulate Multiple (High-albedo) Scattering?*

- Could use pre-integrated tables
  - Need to change for each move in observer position or light position

### *Approximation of Isotropic Particle Scattering*

- Only dependent on light direction

## Issues for Real-time Volumetric Gases: Shadowing

### *How to Compute Real-time Shadows?*

- 2D real-time shadow mapping
  - Only would work for shadowing onto objects, not self-shadowing
  - Problem with transparent objects
- Could create 3D shadow table using texture sliced renderer from direction of eye point
  - Cuts frame rate approximately 25-50% depending on accuracy desired
- Projected imposters to form shadow texture (Dobashi 2000)

## What's Now Available for PC Graphics?

**3D Textures - (e.g., ATI, 3dfx, Nvidia, X-box )**

**Programmable Vertex Shading (e.g., GeForce2, GeForce3)**

**Dependent Texture Reads (e.g., ATI Radeon, GeForce3)**

**Programmable Pixel Shading (e.g., GeForce3)**

## What's Now Available for PC Graphics?

**Stanford Real-Time Programmable Shading Language (Mark, Proudfoot, Hanrahan)**

- Great for real-time programmable shader development and volume shading design
- Re-targetable compiler to optimize passes through graphics pipeline
- Between OpenGL and Renderman



## Hardware Issues With New Advances

### *How Much Flexibility in the New Programmability?*

- Can you add, subtract, multiply, divide?
- Are conditionals allowed?
- How big is the temporary storage?
  - Can you do noise tables?
- Can you use 3D textures just like 2D textures in dependent reads?
- Any order of operations imposed by the hardware (implementation gotcha)?
- What operations are allowed in each part of the pipeline?

## Hardware Issues With New Advances (cont.)

### *What Is the Range of the Values for Each Operation?*

- 0 to 255, -255 to 255, fixed point, float

### *What Is the Precision?*

- 8-bit, 9-bit, 12 bit, 16 bit?
  - Affects complexity of operations that can be performed before quantization errors are visible
- How does the precision vary at different stages of pipeline?
  - E.g., GeForce 3 pixel shaders are floating point, but textures are 8-bit and combiners are 9-bit

## Conclusion

*Procedural Modeling and Animation is :*

**Powerful**  
**Flexible**  
**Extensible**

## Conclusion

### *Important Aspects*

- Flexible volume modeling system
- Accurate illumination and shadowing

### *Procedural Modeling*

- Particle systems, L-systems, blobs can be included
- Flexible, turbulent volume modeling

## Conclusion

### *Volumetric Procedural Implicit Cloud Modeling*

- Ease of control and specification of implicits
- Smooth blending
- Natural appearance from turbulence simulation
- Procedural abstraction
- Parametric control

## Conclusion

### *Real-time Gases Are On the Horizon*

- Latest programmability and capabilities of PC hardware enables a vast array of techniques
- Procedural techniques are well suited for new hardware
  - Eliminate the data transfer bottleneck

### *Future Goal*

- Download procedural cloud to GPU and generate geometry and render on the fly



## Acknowledgements

### *Collaborators:*

- RTSL: Bill Mark, Kekoa Proudfoot, Pat Hanrahan
- Klaus Engel, Rick Parent, Steve May
- Students: Marlin Rowley, Vlad Korolev, Ruchigartha

*Funding: NSF, NASA, DoD, Electronic Arts*